WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, April 27, 2005

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L24	l21 same (frame near5 (table or list$4))	14
☐	L23	l21 and (frame near5 (table or list$4))	23
☐	L22	(creat$4 near5 descriptor) same ((based or depend$4 or accord$4) near5 frame)	7
☐	L21	(creat$4 near5 descriptor) same (based or depend$4 or accord$4)	217
☐	L20	(creat$4 near5 (transaction near2 descriptor))	5
☐	L19	(determin$4 near3 start$4 near3 micro-frame)	2
☐	L18	l7 and l16	10
☐	L17	l14 same L15	12
☐	L16	l14 and L15	91
☐	L15	((plurality or multiple) near3 data near3 structures)	3389
☐	L14	((plurality or multiple) near3 transactions)	4854
☐	L13	l7 and L12	5
☐	L12	l9 or L10	3433
☐	L11	718/100,101,102.ccls	0
☐	L10	713/502.ccls.	596
☐	L9	710/6,52,305,310.ccls.	2853
☐	L8	L7 same schedul$4	6
☐	L7	iTD	412
☐	L6	(non-initialized adj transaction)	3
☐	L5	(initialized adj transaction)	15
☐	L4	(isochronous near3 transaction near3 descriptor)	11
☐	L3	(isochronous adj transaction adj descriptor)	9
☐	L2	(generat$4 near3 transaction near3 schedule)	23
☐	L1	6721815.pn.	2

END OF SEARCH HISTORY

L3: Entry 2 of 9 File: USPT Apr 27, 2004

DOCUMENT-IDENTIFIER: US 6728801 B2
TITLE: Method and apparatus for period promotion avoidance for hubs

Detailed Description Text (7):
FIG. 4 illustrates a typical structure layout of a queue head. Queue head horizontal link pointer (QHLP) 410 comprises four fields: QHLP field 411 contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively. Field 412 is reserved, and bits 4:3 must be written as Os. Field 413 comprising bits 2:1, indicates to the hardware whether the item referenced by the link pointer is a isochronous transaction descriptor (iTD), split transaction isochronous transaction descriptor (siTD) or a queue head. Field 413 allows the HC to perform the proper type of processing on the item after it is fetched. Field 14, bit 0, is the terminate field. If the queue head is in the context of the periodic list, a set (1) bit in field 414 indicates to the HC that this is the end of the periodic list. This bit, however, is ignored by the HC when the queue head is in the asynchronous schedule.

CLAIMS:

1. A method comprising: determining whether a queue head has one of less than a predetermined packet size and equal to the predetermined packet size and whether a period is one of greater than and equal to a predetermined schedule window; storing contents of a current entry in a frame list in a next pointer in the queue head; and replacing the current entry in the frame list with a pointer to a new queue head, wherein a plurality of queue heads are directly coupled to the frame list before any split-isochronous transaction descriptors where split-isochronous transaction descriptors are supported.

6. An apparatus comprising a machine-readable medium containing instructions which, when executed by a machine, cause the machine to perform operations comprising: determining whether a queue head has one of less than a predetermined packet size and equal to the predetermined packet size and whether a period is one of greater than and equal to a predetermined schedule window; storing contents of a current entry in a frame list in a next pointer in the queue head; and replacing the current entry in the frame list with a pointer to a new queue head, wherein a plurality of queue heads are directly coupled to the frame list before any split-isochronous transaction descriptors where split-isochronous transaction descriptors are supported.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L3: Entry 4 of 9 File: USPT Feb 19, 2002

DOCUMENT-IDENTIFIER: US 6349354 B1

** See image for Certificate of Correction **

TITLE: Method to reduce system bus load due to USB bandwidth reclamation

CLAIMS:

7. The system of claim 6 wherein the delay transaction descriptor is an isochronous transaction descriptor.

10. The system of claim 7 wherein the isochronous transaction descriptor is retriable.

Previous Doc Next Doc Go to Doc#

L3: Entry 5 of 9 File: USPT Aug 3, 1999

DOCUMENT-IDENTIFIER: US 5933611 A
TITLE: Dynamic scheduler for time multiplexed serial bus

Detailed Description Text (40):
First, before the USB driver 214 enables the host controller hardware 150 for operation, the host controller driver 214 copies the pointers to the heads of each of the four endpoint descriptor lists from main memory 108 into the respective endpoint descriptor pointer registers in the control registers 802. After being enabled, the serial interface engine 804 finds the first (next) transaction descriptor for the current isochronous endpoint descriptor by causing the PCI interface 810 to master the PCI bus 116 and read, from the current isochronous endpoint descriptor in main memory 108, the field which contains the pointer to the first isochronous transaction descriptor for the current endpoint descriptor. The serial interface engine 804 also at this time extracts the target address and endpoint number from the current isochronous endpoint descriptor in main memory 108 and writes them into the isochronous transaction register in control registers 802. In step 506, the serial interface engine 804 determines that there are no more transaction descriptors for the current isochronous endpoint descriptor by detecting a null value in this field. If the field is not null, then the serial interface engine 804 writes the pointer to the first transaction descriptor for the current isochronous endpoint into the appropriate pointer register in control registers 802.

Detailed Description Text (41):
To accomplish steps 508 and 512, the serial interface engine 804 again masters the PCI bus to read from the isochronous transaction descriptor now pointed to by the isochronous transaction descriptor pointer register in control registers 802. Among other things, the engine extracts the system memory resident transmit/receive data buffer start location and writes it to the isochronous transmit/receive data buffer pointer register in control registers 802, and sets the valid bit. It also extracts the transaction direction, byte count and transaction speed for the current isochronous transaction, and writes them into the isochronous transaction register in control registers 802. The engine then reads the field containing the frame number for the current isochronous transaction and stores it in the "first frame number" register in control registers 802. The host controller hardware 150 now has both the current frame number in one of the counters 912, and the frame number designated for the current isochronous transaction descriptor in the appropriate control register 802, and can compare them to accomplish steps 508 and 512.

Detailed Description Text (42):
If the algorithm reaches step 514, in which the current isochronous transaction is to be executed, then this step is performed by the dynamic scheduler 806 upon command from the serial interface engine 804. If the dynamic scheduler 806 is not yet ready to receive such a command, then the serial interface engine 804 stalls until it is. Once the dynamic scheduler 806 accepts the command, the serial interface engine 804 does not need to wait for the transaction to complete over the USB; rather, it proceeds with step 504 by gain mastering the PCI bus to extract the pointer, in the current isochronous transaction descriptor, which points to the next isochronous transaction descriptor or null).

First Hit Previous Doc Next Doc Go to Doc#

Generate Collection Print

L3: Entry 6 of 9 File: DWPI Jun 10, 2004

DERWENT-ACC-NO: 2004-486608
DERWENT-WEEK: 200446

TITLE: Apparatus for improving performance of enhanced host controller interface for universal serial bus device e.g. printer, has USB host controller that couples queue heads to frame list before split-isochronous transaction descriptors

Basic Abstract Text (1):
NOVELTY - A universal serial bus (USB) 2.0 host controller (HC) couples several queue heads to a HC frame list, before any split-isochronous transaction descriptors (siTD) in the USB 2.0 system. The queue heads are coupled to the HC frame list before or after initialization of the host controller.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L8: Entry 5 of 6 File: USPT Jul 2, 1985

DOCUMENT-IDENTIFIER: US 4527250 A

**** See image for Certificate of Correction ****

TITLE: Video computer terminal with detachable intelligent keyboard module

Detailed Description Text (37):

The executive program 150 represented in FIG. 14 is a REX-80 executive program, which is commercially available from Systems and Software, Inc., Downers Grove, Il. Besides the executive program 150, the vendor provides instructions for the development of specialized program units called tasks. Each task is a program in itself that directs all of the functions of the microprocessor 68 during a particular time in which the task is in an active or running state. The executive program 150 uses certain parameters, referred to as initial task descriptors (ITD's), which are provided by the task originator and which are used to build up task control blocks (TCB's). Thus a task can be represented in a linked list or queue by an address of a task control block containing the unique information characterizing the particular task. One item of information is the priority of the task; another important item of information is an event control byte or a word. This event control byte or word is based upon an event flag that signals a precondition determining whether the task is ready to be scheduled for the running or active state.

Previous Doc Next Doc Go to Doc#

L13: Entry 3 of 5 File: USPT May 28, 2002

DOCUMENT-IDENTIFIER: US 6397243 B1
TITLE: Method and device for processing several technical applications each provided with its particular security

Detailed Description Text (6):
For example, at the end of time slot T2 assigned to technical application P2, a start interrupt ITD3 is generated, it instructs the starting of the next technical application P3. This next technical application P3 starts safely at a fixed instant in the cycle which corresponds to the end of slot T2 so that the duration of time slot T3 which is assigned to technical application P3 is not modified. While the computer is operating, a start interrupt ITD causes the computer to divert to another technical application with a view to its processing. The start interrupts are referenced ITD1, ITD2, ITD3, ITD4.

Detailed Description Text (8):
To avoid this, provision may be made to precede a start interrupt ITD with an end interrupt ITF which interrupts the current technical application, unless an atomic section is currently being executed. The technical application in question will then be interrupted only at the end of the atomic section. This end interrupt ITF can be temporarily masked by the basic software since it is only active in certain cases, when it arrives in the absence of any step which may not be interrupted. The end interrupts are referenced ITF1, ITF2, ITF3, ITF4.

Detailed Description Text (11):
The time interval between an end interrupt IF and the start interrupt ITD which follows it is greater than or equal to the longest duration of the atomic sections of the basic software. This time interval is, for example, of the order of 500 .mu.s.

Detailed Description Text (15):
The position of the cycle interrupts ITRTC, of the start ITD and end ITF interrupts are programmable, preferably with a resolution of at least 50 microseconds.

Detailed Description Text (18):
The device also comprises means for defining, during the work time cycles, the time slots allotted to the technical applications and which generate, at the end of a time slot, the start interrupt ITD. These means can comprise for each technical application processed during a time slot (in FIG. 2 the means used for application P2 of rank 2 have been represented), a start duration register RITD2 and an equality comparator C'2 and call upon the duration counter CT. The start duration register RITD2 makes it possible to define the duration between the resetting to zero of the time counter CT and the start of the next technical application. The equality comparator C'2 compares the value of the duration counter CT with that of the start duration register RITD2 and in case of equality generates a start interrupt ITD2.

Detailed Description Text (41):
FIG. 7 shows a partial example of a device for processing several technical applications in accordance with the invention. The various applications are referenced P0 to P7. Their code in the entitlements register RD is mentioned for

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L20: Entry 3 of 5 File: USPT Sep 4, 2001

DOCUMENT-IDENTIFIER: US 6286011 B1

** See image for Certificate of Correction **

TITLE: System and method for recording transactions using a chronological list superimposed on an indexed list

CLAIMS:

2. The method of claim 1, wherein the step of creating a new transaction entry for the transaction comprises:

creating a transaction descriptor field comprising a transaction descriptor for identifying the transaction;

creating a time stamp field comprising a time identifier for indicating a time when the transaction occurred;

creating a chronological list pointer field comprising a chronological list pointer for identifying a subsequent transaction entry corresponding to a transaction received after the new transaction; and

creating an indexed list pointer field containing an indexed list pointer for identifying a next transaction entry corresponding to the same index as the new transaction entry.

Previous Doc Next Doc Go to Doc#

L20: Entry 4 of 5 File: USPT Jul 4, 2000

DOCUMENT-IDENTIFIER: US 6085200 A
TITLE: System and method for arranging database restoration data for efficient data recovery in transaction processing systems

Detailed Description Text (23):
FIG. 5 illustrates a linked-list structure comprising multiple transaction descriptors. The transaction descriptors are data structures which manage changes to the database, audit trail or other data retention structures on a per-transaction basis. A number of these transaction descriptors can be predefined, and any additional transaction descriptors required can be created by the I/O task management system when needed.

Detailed Description Text (24):
Each of the transaction programs in the computer system is associated with a transaction descriptor, which, in one embodiment of the invention, is created by scheduling its corresponding transaction program. The transaction descriptors illustrated in FIG. 5 include transaction descriptor A 500, transaction descriptor B 502, through transaction descriptor n 504. In one embodiment of the invention, each transaction descriptor includes a plurality of memory fields. For example, transaction descriptor A 500 includes a control information field 506, and audit data pointer field 508, a database retention pointer field 510, a source message pointer 512, and a destination message pointer 514. Transaction descriptors B 502 through transaction descriptor n 504 include analogous memory fields.

Detailed Description Text (28):
Transaction descriptors make the creation of the audit trail more efficient. Because the transaction descriptors are stored in a non-volatile storage medium, the audit data need not be written immediately to the audit trail at transaction commit/rollback to protect the data. Instead, the audit data can be queued within the non-volatile memory. This increases system throughput, as a host can begin a new transaction without waiting for the audit trail data to be recorded. Additionally, the use of transaction descriptors minimizes the amount of data that must be written to the audit trail, because the transaction identifier need only be recorded once for the data included in the transaction descriptor, and need not be repeated for every update associated with the transaction.

Detailed Description Text (30):
The source message pointer field 512 includes an identifier that designates the location of source messages received by way of scheduling a particular transaction program by step control. Therefore, when a source message is sent via a host processor to the I/O task management system 216, the source message itself is stored in the source message data structure 520, and a pointer or link identifying the location of the message data is stored in the source message pointer field 512 as part of the transaction descriptor A 500 when the transaction program which is providing the transaction has been scheduled. More particularly, the pointer is stored first in the input message queue 406 of FIG. 4, and when the transaction is scheduled, the transaction descriptor A 500 is allocated (or created if necessary) and the source message pointer is moved to the source message pointer field 512.

Detailed Description Text (36):

At any point in time during the life of a transaction program, the transaction descriptors contain all of the information necessary to recover a particular transaction. A transaction descriptor is marked as "active" when the transaction descriptor is allocated/created. In the event that a transaction does not reach a stable point, the transaction issues a rollback request 603. The control information is updated 604 to reflect the transaction state as being rollback. If a host or component failure occurs, this state (e.g., either rollback or active) indicates to a recovery module that the transaction must be rolled back, since the data has not been committed. The retention data temporarily stored in the database retention memory structure 518 (FIG. 5) is copied back to the memory cache to negate the transaction update requests, as indicated at step 606. The destination messages may be discarded 608 (or alternatively subsequently overwritten), and the source messages are requeued 610 for subsequent rescheduling of the transaction program by moving the source message pointer in the source message pointer field 512 (FIG. 5) back into the input message queue 406 (FIG. 4). The audit data is updated 612 to indicate a transaction rollback event, and the audit data is queued 614 for subsequent writing to the audit trail by moving the audit data pointer in the audit data pointer field 508 (FIG. 5) to a queue designed to sequentially store the audit data in the audit trail. The transaction descriptor is then released 615 for use by other transactions.

Previous Doc Next Doc Go to Doc#